Amendments to the Specification:

Please delete paragraph [0010]

[0010] ~~In still yet another aspect, the invention features a computer data signal embodied in a carrier wave for use with a computer system having a display and capable of generating a user interface through which a user may interact with the computer system. The computer data signal comprises program code for providing a service provider interface (SPI) defining a plurality of procedures for communicating with a meeting services application of the online meeting system, program code for implementing one of the procedures of the SPI by a software module to perform, when executed, a meeting-related operation customized in accordance with the third party system, and program code for calling by the meeting service application the implemented SPI procedure of the software module to perform the meeting related operation.~~

Please amend paragraph [0020] to read as follows:

[0020] The SPI 116 provides a plurality of methods for invoking a service provider. To be compliant with the online meeting services application 104, each module 112 implements each method named below. The actual function performed by each particular method is customized to the needs of the third-party system 108. The methods include (using Java-like pseudo code):

Please amend lines 5 and 12 of paragraph [0023] to read as follows:

[0023] The other methods in the SPI 116 relate to particular events that are about to occur or have just occurred with respect to a meeting. In one embodiment, the events include creating a meeting 118, editing a meeting 120, deleting a meeting 124, and performing a meeting state change 128. Calls to

these methods from the meeting services application 104 serve to notify the called module 112 of these events. The methods pertaining to these events are categorized into one of two types: "pre-" methods and "post-" methods. Accordingly, each event is invoked in two phases: pre-commit and post-commit. Changes do not occur (i.e., are not committed) upon procedure calls to the "pre" methods. Rather, modules 112 use "pre" methods to prepare for an occurrence of an event. Changes occur upon procedure calls to the "post" methods. That is, for each proposed or required event, the meeting services application 104 invokes the corresponding "pre" method before the event is finalized, confirmed, or committed and invokes the corresponding "post" methods after the event has been finalized. The dividing of events into two phases makes the aggregation of modules 112 act as a two-phased commit system. At either of the two phases, each module 112 can alter the meeting process. In FIG. 2, pre-methods are identified as doubled-headed arrows because each pre-commit method can throw an exception to the meeting services application 104, to prohibit or deny the request for the particular event called for by that method. In an alternative embodiment, one or more of the pre-commit methods return an error instead of throwing an exception.

Please amend paragraph [0029] to read as follows:

[0029] The state methods 128 are particularly useful for a third-party system concerned with tracking the progress of a meeting and performing actions in response to a meeting's current state. Whenever the state of a meeting is about to change, the ~~meetings~~ meeting services application 104 calls the preStateChange() method of each installed module 112. After the transition occurs, the meeting services application 104 calls to the postStateChange() method. The ~~meetings~~ meeting services application 104 maintains a

respective state machine 106 for each meeting to track the progress of that meeting.

Please amend paragraph [0030] to read as follows:

[0030] For example, consider a module 112 designed to perform customer billing. For the purpose of this example, consider that a bill corresponds to the amount of time a particular customer's meeting stays online. Accordingly, the billing module 112 is interested in state changes associated with the online meeting. When the meeting services application ~~program~~ 104 invokes the postStateChange() method of the billing module, the code of the postStateChange() method reviews the meeting object and the state object. If the state transition indicates that the meeting has progressed from SCHEDULED to ACTIVE, the billing system can establish a start time for the activation of the meeting. Subsequently, when the meeting transitions from the ACTIVE state to ENDED state, the billing system can establish an end time. The billing module 112 then uses its own resources to calculate the bill for the customer based on the amount of time used.

Please amend lines 5 and 9 of paragraph [0032] to read as follows:

[0032] After launching the meeting services application 104, the user can choose (step 208) to perform any of the meeting services (i.e., creating, editing, deleting, or running a meeting). Note, the running of a meeting can occur automatically at the scheduled time, that is, the user need not choose to run the meeting. The meeting services application 104 receives (step 212) the selection of the user. Selection of any one of the creating, editing, and deleting service events causes the meeting services application 104 to invoke (step 216) the corresponding "pre-" method of the SPI 116 of each installed module 112. If the user's choice is to start a meeting, the meeting services application 104

invokes (step 220) the preStateChange() method of the SPI 116 of each installed module 112 upon detecting that the meeting is transitioning from one defined state to another.

Please amend paragraph [0035] to read as follows:

[0035] The process 200 described in FIG. 3 is not intended to limit the invention to any particular order of events. For example, step 204 can occur concurrently with any of other steps in the process 200. Further, the process 200 can occur multiple times for any given meeting. For example, consider that at step 208 the user chooses to create a meeting that starts at a later time. Assume, for this example, that no exceptions are thrown. At step 216, the ~~meetings~~ meeting services application 104 invokes the preCreate() method and, at step 236, the postCreate() method. (Mention of the other performed steps is omitted to simplify this example.) Later, when the meeting subsequently starts at the scheduled time, the ~~meetings~~ meeting services application 104 invokes the preStateChange() method at step 220 (i.e., state of the meeting transitions from SCHEDULED to ACTIVE) and the postStateChange method at step 236. Then when the meeting ends, the meeting services application 104 invokes the preStateChange() and postStateChange() methods again (i.e., state of the meeting transitions from ACTIVE to ENDED).

Please amend paragraph [0041] (previously amended in the response to the non-final office action) to read as follows:

[0041] The present invention may be implemented as one or more computer-readable software programs embodied on or in one or more computer-readable storage medium. The computer-readable storage medium can be, for example, any one or combination of a floppy disk, a hard disk, hard-disk drive, a CD-ROM, a DVD-ROM, a flash memory card, an EEPROM,

an EPROM, a PROM, a RAM, a ROM, or a magnetic tape. In general, any standard or proprietary, programming or interpretive language can be used to produce the computer-readable software programs. Examples of such languages include C, C++, Pascal, JAVA, BASIC, Visual Basic, and Visual C++. The software programs may be stored on or in one or more computer-readable storage medium as source code, object code, interpretive code, or executable code.